情報工学実験Ⅲレポート

C言語によるアフィン変換の実装

15X3034 音賀　優颯

wasabi

1. 目的

　プログラミング言語により，アフィン変換を実装することで，アルゴリズムの実装を体験するとともに，理解を深める．

1. 内容

　C言語を用いて任意の拡大率，回転角を与えることで，拡大，回転された画像が保存されるプログラムを作成した．また，出力する画像の各画素はバイリニア補間を用いた．

1. コード

以下に作成したコードを示す.

image\_process.c

1 #include <stdio.h>

2 #include <stdlib.h>

3 #include <string.h>

4 #include <math.h>

5

6 typedef struct{

7 char\* magic[3];

8 int row,column,max;

9 unsigned char\*\*\* array;

10 } image\_t;

11

12 int read\_image(char\* input, image\_t\* img)

13 {

14 int i,j,k;

15 char\* buffer;

16 FILE\* fp;

17

18 buffer=calloc(BUFSIZ,sizeof(char\*));

19 sprintf(buffer,"./%s",input);

20 if((fp=fopen(buffer,"rb"))==NULL){

21 fprintf(stderr,"入力ファイルが開けません.\n");

22 exit(1);

23 }

24

25 //read img

26 for(i=0;i<3;i++){

27 fgets(buffer,BUFSIZ,fp);

28 if(i==0)

29 sprintf(img->magic,"%s",(strtok(buffer,"\n")));

30 if(i==1){

31 img->row=atoi(strtok(buffer," "));

32 img->column=atoi(strtok(NULL," "));

33 }

34 if(i==2)

35 img->max=atoi(strtok(buffer," "));

36 }

37 //memoryの動的確保

38 img->array=malloc(3\*sizeof(unsigned char\*\*));

39 for(i=0;i<3;i++){

40 img->array[i]=malloc(img->column\*sizeof(unsigned char\*));

41 for(j=0;j<(img->column);j++){

42 img->array[i][j]=malloc(img->row\*sizeof(unsigned char));

43 }

44 }

45 //読み込み

46 for(j=0;j<(img->column);j++){

47 for(k=0;k<(img->row);k++){

48 fread(&(img->array[0][j][k]),sizeof(unsigned char),1,fp);

49 fread(&(img->array[1][j][k]),sizeof(unsigned char),1,fp);

50 fread(&(img->array[2][j][k]),sizeof(unsigned char),1,fp);

51 }

52 }

53

54 return 0;

55

56 }//end of read\_image

57

58 int affine(double scaler, double theta, image\_t\* img\_in, image\_t\* img\_out)

59 {

60 int i,j,k;//for

61 double sn,cs;//sin,cos

62 sn=sin((theta/180)\*M\_PI);

63 cs=cos((theta/180)\*M\_PI);

64

65 int in\_row,in\_column;//入力画像のサイズ

66 in\_row=img\_in->row;

67 in\_column=img\_in->column;

68

69 int out\_row,out\_column;//出力画像のサイズ

70 double ix,iy;//出力画像の画素から計算した元画像の座標

71 int nx,ny;//出力画像の画素から計算した元画像の座標

72

73 //出力画像のサイズを計算

74 if((theta>=0 && theta<=90) || (theta>=180 && theta<=270)){

75 out\_row=abs(in\_row\*cs)+abs(in\_column\*sn);

76 out\_column=abs(in\_column\*cs)+abs(in\_row\*sn);

77 }

78 else{

79 out\_row=abs(in\_column\*cs)+abs(in\_row\*sn);

80 out\_column=abs(in\_row\*cs)+abs(in\_column\*sn);

81 }

82 out\_row\*=scaler;

83 out\_column\*=scaler;

84

85 //メモリの動的確保

86 img\_out->array=malloc(3\*sizeof(unsigned char\*\*));

87 for(i=0;i<3;i++){

88 img\_out->array[i]=malloc(out\_column\*sizeof(unsigned char\*));

89 for(j=0;j<out\_column;j++){

90 img\_out->array[i][j]=malloc(out\_row\*sizeof(unsigned char));

91 }

92 }

93

94 //画像の中心を軸にして回転させるため,画像サイズの半分ずらす

95 for(i=-out\_column/2;i<out\_column/2;i++){

96 for(j=-out\_row/2;j<out\_row/2;j++){

97

98 //回転後の座標を計算

99 //拡大率でわると元画像が内接する長方形のサイズになる

100 ix=(j\*cs-i\*sn)/scaler;

101 iy=(j\*sn+i\*cs)/scaler;

102

103 //画像の中心が元画像の中心と

104 //同じになるように元画像の半分戻す

105 ix+=in\_row/2;

106 iy+=in\_column/2;

107

108 //倍精度をint型にキャスト

109 nx=ix;

110 ny=iy;

111

112 //ix,iyはバイリニア補間の重みになる

113 ix=ix-nx;

114 iy=iy-ny;

115

116

117 if(nx>=0 && ny>=0 && nx<in\_row-1 && ny<in\_column-1){

118 for(k=0;k<3;k++){

119 img\_out->array[k]\

120 [i+out\_column/2][j+out\_row/2]=(int)(\

121 (double)img\_in->array[k][ny][nx]\*(1-ix)\*(1-iy)+\

122 (double)img\_in->array[k][ny][nx+1]\*ix\*(1-iy)+\

123 (double)img\_in->array[k][ny+1][nx]\*(1-ix)\*iy+\

124 (double)img\_in->array[k][ny+1][nx+1]\*ix\*iy);

125 }

126 }

127 //元画像が値を持っていない（元画像の外側を参照している）

128 else{

129 img\_out->array[0][i+out\_column/2][j+out\_row/2]=150;

130 img\_out->array[1][i+out\_column/2][j+out\_row/2]=150;

131 img\_out->array[2][i+out\_column/2][j+out\_row/2]=150;

132 }

133 }

134 }

135

136 //header部分

137 sprintf(img\_out->magic,"%s",img\_in->magic);

138 img\_out->row=out\_row;

139 img\_out->column=out\_column;

140 img\_out->max=img\_in->max;

141

142 return 0;

143 }//end of affine

144

145 int make\_image(char\* out, image\_t\* img)

146 {

147 int i,j;

148 char\* buffer;

149 FILE\* fp;

150

151 buffer=calloc(BUFSIZ,sizeof(char\*));

152 sprintf(buffer,"./%s",out);

153 if((fp=fopen(buffer,"wb"))==NULL){

154 fprintf(stderr,"出力ファイルが開けません.\n");

155 exit(1);

156 }

157 fprintf(fp,"P6\n");

158 fprintf(fp,"%d %d\n",img->row,img->column);

159 fprintf(fp,"%d\n",img->max);

160 for(i=0;i<img->column;i++){

161 for(j=0;j<img->row;j++){

162 fwrite(&img->array[0][i][j],sizeof(unsigned char),1,fp);

163 fwrite(&img->array[1][i][j],sizeof(unsigned char),1,fp);

164 fwrite(&img->array[2][i][j],sizeof(unsigned char),1,fp);

165 }

166 }

167 return 0;

168

169 }//end of make\_image

170

171 int main(int argc, char\*\* argv)

172 {

173 int i;

174 image\_t img\_in,img\_out;

175

176 if(argc<5){

177 fprintf(stderr,"引数が足りません,入力ファイル \

178 出力ファイル 拡大率 回転角 の順で実行してください.\n");

179 exit(1);

180 }

181

182 read\_image(argv[1],&img\_in);

183 affine(atof(argv[3]),atof(argv[4]),&img\_in,&img\_out);

184 free(img\_in.array);

185 make\_image(argv[2],&img\_out);

186 free(img\_out.array);

187

188 return 0;

189 }//end of main

1. 結果

元の画像をFig. 1に元画像として示す．



Fig. 1　元画像

以下のように実行した結果をFig. 2に出力画像（2.4,30）として示す．

$image\_process lena.ppm modified\_lena1.ppm 2.4 30



Fig. 2　出力画像（2.4,30）

以下のように実行した結果をFig. 2に出力画像（0.6,-50）として示す．

$image\_process lena.ppm modified\_lena\_down.ppm 0.6 -50



Fig. 3　出力画像（0.6,-50）

1. 考察

関数affine()について

* 1. 出力サイズの計算

　90度回転させるごとに，画像の横と縦の長さが入れ替わると考えて，回転角thetaが0-90の時，180-270の時とそれ以外の角度の時で計算方法を別にした．最後に拡大率をかけることで，任意の拡大率ので拡大した画像を任意の角度回転させたが内接する長方形を作成することができる．

* 1. 画像の回転

　変換式を適用する前に座標を出力画像のサイズの1/2だけマイナス方向へ，シフトした．これは，画像の中心を軸（原点）にして回転させるためである．そのまま変換式を提要すると，画像の左が軸になって回転してしまう．シフトの距離と拡大率で除算した理由，入力画像のサイズの1/2だけシフトした理由は次のように考えた．出力画像のサイズはすでに回転した画像が内接する大きさなので，そこで回転theta分だけ逆回転させると，拡大のみ行った画像の外側に値を持たないような画像を仮想的に考えることができる．これを拡大率で除算すると中央にある画像は入力画像と同じサイズになるので，入力画像のサイズの1/2だけシフトすることで求める座標が得られる．

　また変換式は以下のように求められた．

回転前の座標を()，回転後の座標を()，原点からの距離をr，回転角をとするとそれぞれ

と表せる．画像のy軸は一般的な直行座標と逆向きなので，マイナスをかけて考慮する．ここで，加法定理より，

ただしプログラム内で用いた処理では逆回転なのでで再計算すると,sinの値の正負が反転するため，最終的に求める()は次のようになる．

* 1. バイリニア補間

　上記によって求めた座標は浮動小数点数であるため，４近傍の画素を用いて画素を計算した，浮動小数点で求めた座標ix,iyをint型にキャストすることで得られるnx,nyは常にix,iyを囲む４近傍の点の左上とすることができる．また，ix，iyとnx,nyとの差は距離と同値なので，それを考慮して４近傍の画素に対して重みをかけることができる．最後に計算された画素値をint型にキャストすることでバイリニア補間によって計算された出力画像の画素を得ることができた．

1. 結論

　プログラミング言語により，アフィン変換を実装することで，アルゴリズムの実装を体験するとともに，理解を深められた．